IN THE SPECIFICATION:

Kindly amend the specification as noted below.

[0010] ~~**Figure**~~ **Figures 6A-6C** ~~illustrated~~ illustrate header information for the IP, TCP and UDP protocols respectively.

[0020] **Figure 1** is a diagrammatic representation of an exemplary network environment 10 within which the present invention may be deployed to perform application performance analysis and, more specifically, to calculate network latencies and to output such latencies, for example, as a graphical display. The network environment 10 is shown to include a client machine 12 that hosts a client application 13. In an exemplary embodiment, the client application 13 communicates with a server application ~~17~~ 19, hosted on a server machine 16, via a network 14. The network 14 may be any one of a number of well-known network types, such as an intranet, the Internet, a Wide Area Network (WAN) or a Local Area Network (LAN) and may employ any one or more of a number of well-known communication protocols (e.g., IP, TCP, UDP etc.). The client machine 12 is further shown to host a data capture application 18 (e.g., a sniffer or program protocol analyzer) that generates a trace file 20. Commercially available examples of such a data capture application 18 include the Microsoft Network Monitor (NetMon) developed by Microsoft Corp. of ~~the~~ Redmond, Washington and the Optimal Smart Agent developed by the Optimal Networks Corp. of Mountain View, California, now owned by the Compuware Corp. of Detroit Michigan. The trace file 20 generated by the data capture application 18 includes an entry for each packet observed by the data capture application 18 to be transmitted from, or received at, the client machine 12. While an entry within the trace file 20 may include any data contained in a packet [[to]] that the data capture application 18 is configured to capture, for the purpose of the present invention, each entry includes at least (1) a packet identifier, preferably comprising unique, static content extracted from the packets, and (2) a timestamp indicating a time at which the data capture application 18 observed a specific packet. **Figure 5** provides an illustration of an exemplary trace file 20 that includes entries for N packets, each entry recording a packet identifier 30, a timestamp 32, a packet size 34, and packet data 36, which may include header information. The packet identifier 30 is determined by the data capture application 18 so as to be highly unique (although not necessarily absolutely unique) and static for a packet observed by the application 18. The exact content of the packet identifier may vary from protocol to protocol. For example, Internet Protocol (IP) packets may have a packet identifier 30 comprised from the IP source and destination addresses, an IP fragment identifier, and an IP fragment offset. TCP packets, on the other hand, may have packet identifiers 30 composed of TCP ports, a TCP sequence and/or TCP acknowledge numbers in addition to IP fields. UDP packets may have packet identifier 30 composed of UDP ports used in conjunction with IP fields.

[0036] At block 82, the correlation engine 52 determines the transmission direction of packets that are common to both the primary and secondary trace files, utilizing ~~with~~ any of the methodologies discussed above.

[0046] At decision block ~~112~~ 122, a determination is made as to whether clock drift is exhibited by the calculated offsets. Referring to **Figure 9A**, a series of calculated offsets is illustrated that exhibited acceptable time variances and a consistency across multiple unique packets, this consistency being indicative of an absence of clock drift. **Figure 9B**, on the other hand, indicates a trend of increasing offsets for unique packets, this trend being indicative of clock drift. It will be appreciated that the success of the synchronization method 64 is dependent upon the accuracy of the timestamps, within the correlated list 72 and as extracted from the multiple trace files.

[0054] In addition to the clock drift and fragmentation issues discussed above, in further embodiments, a number of other issues may be addressed by the present invention. First, the merge module 42 may utilize different protocols to limit a search space within a trace file and may utilize actual data in the datagram of a packet as an additional correlation point when the potential for incorrect correlation exists. As discussed above, certain fields from IP and PC packets may be utilized to correlate packets. These may be reused by the TCP stack in a number of ways. For example, IP addresses may be used by different nodes if DHCP is implemented, TCP and UDP ports may be reused by subsequent connections, IP fragment identifiers repeat after 65536 packets have been sent, and TCP sequence and acknowledge numbers will repeat after 4294967296 bytes of data have been sent. For TCP packets, there is [[in]] a very small probability that a correlation, performed in the above-described manner, would be incorrect even if traces are compared from different days of traffic between the same IP addresses and ports. However, simpler protocols (e.g., IP) have much less correlation data to rely on. Specifically, as described above, the merge module 42 may utilize only IP addresses and fragment identifiers to correlate IP packets. Accordingly, if more than 65535 packets of pure IP traffic are transmitted between the same ~~to~~ two nodes, correlation may become unreliable. To address this potential problem, the present invention proposes that if correlated TCP packets are seen in the same trace file, these may be utilized to limit search space for matching IP packets and thereby improved reliability. Additionally, the merge module 42 may utilize actual data in the datagram as an additional correlation point.